in: *Proc. Int. Workshop on New Directions of Control and Manufacturing*, Hong Kong, 1994, pp. 147–154.

A Modeling Method for Flexible Manufacturing Systems based on Colored Petri Nets *

Armin Zimmermann
Institut für Technische Informatik
Technische UniversitAat Berlin

Abstract

This paper addresses the problem of adequate modeling of flexible manufacturing systems (FMS) with Petri nets. Advantages and disadvantages of uncolored and colored approaches are shown with an application example.

Starting from these considerations, we define a formalism which is especially tailored for the modeling of flexible manufacturing systems. With the method proposed in the paper, we intend to combine the profitable properties of both colored and uncolored Petri nets. This is done by the predefinition of color sets for the modeling of parts *and* resource states *of the FMS.*

Finally, the application of the modeling formalism to a manufacturing system example shows the usefulness of our approach.

1 Introduction

Flexible manufacturing systems (FMS) are complex configurations of robots, machines, and transport systems. They can handle many different parts at the same time, and often the variety of products changes. Because of their behavioral complexity, modeling and performance analysis of FMS play a major role for their efficient use.

Stochastic Petri nets [1], the stochastic extension of Petri nets [18], are a promising tool for the description, analysis and implementation of the control of FMS [20]. They represent a graphical description which is an accepted method for the modeling of discrete event systems. Manufacturing systems are one of the oldest application areas of Petri nets. Reference [20] contains a recent survey.

FMS have often been modeled and analyzed in the literature with Petri nets [3, 19], generalized stochastic Petri nets [4, 5], and stochastic Petri nets containing non-exponentially distributed firing times [13, 17, 23].

Zurawski and Dillon [26] examined methods for the systematic construction of typical parts of FMS. They encountered the problem of modeling buffers containing different parts by multiple places in uncolored petri nets. The obtained nets are used for the analysis of functional abstractions.

Colored Petri nets (CPNs) have been used for the modeling and deadlock analysis of FMS [21, 22] and for the real-time control of a manufacturing system at the cell level [15].

In the past, different approaches to the modeling of FMS with specialized Petri nets have been made. Bruno et. al. [7] introduced *Prot nets* as an extension to Petri nets with marking-dependent arc expressions for the modeling of discrete event systems. The use of different types of places in an uncolored net for the generation of live and safe Petri net models was investigated by Beck and Krogh [6]. Gentina and Corbeel examined possibilities of automatic Petri net design [12]. They used artificial intelligence methods to automatically derive a state machine of the described FMS.

The remainder of this paper is organized as follows. In section 2 a cell of a manufacturing system is presented, which is then modeled with an uncolored and a colored Petri net, showing the advantages and disadvantages of both methods. Subsequently, a special modeling method for flexible manufacturing systems is developed and applied to the example in section 3. Finally, some concluding remarks are given.

2 An Example: A Manufacturing Cell

In this section an example of a manufacturing cell is developed. This example will be modeled with different kinds of Petri nets, demonstrating the use of the proposed method.

Flexible manufacturing systems are composed of machines, buffers, transport systems, and a control system. Machines are able to process parts of different types, while buffers are needed for intermediate storage of parts. The transport system has to be able to move parts in the FMS. The control system decides which part is to be processed next by an idle machine. Machines are sometimes grouped into elementary manufacturing cells.

As a modeling example, we use the simple man-

*This work was supported by the German Research Council under grant Ho 1257/4-1.

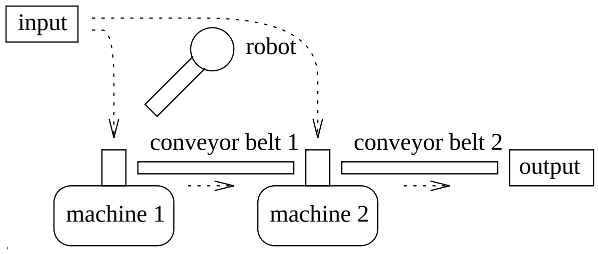

Fig. 1: Layout of the manufacturing system cell

ufacturing cell shown in Fig. 1. The cell consists of two machines, a robot, two conveyor belts and the input and output buffers of the cell. The global transport system stores parts to be processed in the input buffer. The products can then be moved to machine 1 or machine 2 by the robot. When a part is processed by machine 1, it can be moved to machine 2 by the conveyor belt 1. Finished parts leave the cell via the output buffer.

Typically, the production line organisation of an FMS is referred to as a *job shop*. In a *job shop*, there are no predefined routes of parts through the system, rather for each product a production route has to be defined [20]. This property ensures the versatility and flexibility of the manufacturing system. The routes of the two products processed by our manufacturing cell are shown in Fig. 2.

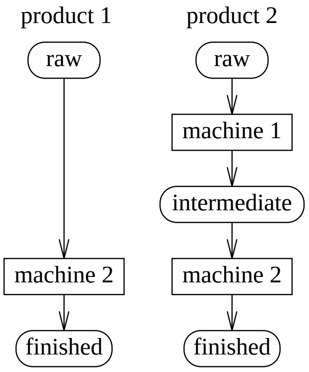

Fig. 2: Production routes of the two products

Products of type 1 enter the cell in state *raw*. After they have been processed by machine 2, they are *finished* and leave the cell. Type 2 products are processed by both machines before they are completed. The state of product 2 parts after leaving machine 1 is called *intermediate*.

2.1 FMS Modeling With Generalized and Stochastic Petri Nets

Generalized and stochastic Petri nets [2] have often and successfully been used for the modeling and performance evaluation of (simple) manufacturing systems [4, 5]. Before we model our example with a generalized and stochastic Petri net, a brief description of the notation is given.

A generalized stochastic Petri net (GSPN) is an eight-tuple [9]:

$$\left(P, T, \Pi, W^-, W^+, W^H, \Lambda, M_0\right) \tag{1}$$

where

P is the set of places,

T is the set of transitions (with $P \cap T = \emptyset$),

Π is the priority function, which maps transitions into nonnegative natural numbers, representing their priority level (for timed transitions, the priority is equal to zero),

W^-, W^+, W^H are the input, output and inhibition functions, which map transitions on multisets (i.e. sets with multiplicity) of places. These values define the multiplicities of the input, output, and inhibitor arcs between transitions and places.

Λ maps transitions into positive real numbers. For timed transitions t_k, $\Lambda(t_k)$ is called the *rate* of the transition (i.e. the firing time of transition t_k is exponentially distributed with parameter $\Lambda(t_k)$). If t_k is an immediate transition (i.e. it fires without delay), $\Lambda(t_k)$ denotes the *firing weight*, which is used to solve conflicts between immediate transitions, and

M_0 denotes the initial marking.

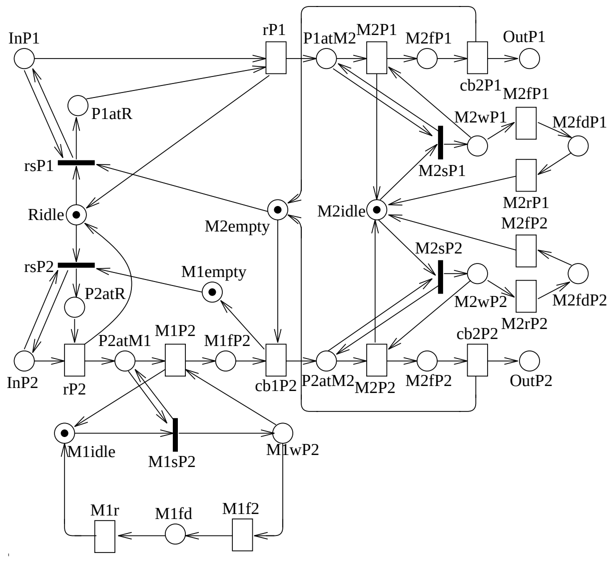

Fig. 3: GSPN model of the manufacturing cell

The GSPN model of the flexible manufacturing cell is shown in Fig. 3. In GSPNs, there are no individual tokens. Therefore, all buffers possibly containing parts of both types (or parts of different states) have to be modeled with two places in order to distinguish between them. Thus, the line of buffers and conveyor belts is unfolded into two paths. Places *InP1* and *InP2* model the input buffer; for instance, a token in *InP1* corresponds to one part of type 1 in the input buffer. Accordingly, places *OutP1* and *OutP2* form the output buffer. Transitions *cb*<i>*P*<k> denote the transport of a part of type <k> by the conveyor belt <i>. Places *M1empty* and *M2empty* ensure that there is at most one part of any type in machine 1 and machine 2, respectively.

Machine 1 is modeled by the subnet below the transport lines. The initial state of the machine is idle (place *M1idle* contains one token). When a part of type 2 arrives at the machine (a token arrives in place *P2atM1*), the machine starts to work (transition *M1sP2* fires and *M1P2* becomes enabled). During the work, the machine can fail (transition *M1f2* fires) and changes its state to failed (*M1fd* contains one token). The machine is repaired after some time has passed (transition *M1r* fires), gets idle and starts to work on the same part immediately again (transition *M1sP2* fires). When the processing of the part is done, *M1P2* fires and place *M1fP2* contains a token representing a part of type 2 finished by machine 1.

Parts of type 2 in intermediate state can then be moved to machine 2 by the conveyor belt 1 (transition *cb1P1* fires). With this firing, place *M1empty* receives a token, and thus a new raw part may be moved to machine 1.

Although the subnet which models machine 2 looks more complicated, it is almost the same subnet as for machine 1, with the difference that it had to be duplicated in order to differentiate between the two part types. Places and transitions of machine 2 are named according to machine 1. Thus, we refer to the explanation of machine 1 for the details.

The state of the robot can be one of the following: idle (a token is in place *Ridle*), moving a part of type 1 to machine 2 (a token is in place *P1atR*), and moving a part of type 2 to machine 1 (a token is in place *P2atR*). The actual transport action is modeled with one of the two transitions *rP1* and *rP2*. When the robot starts to move a part, either transition *rsP1* or *rsP2* fire, thus changing the state of the robot appropriately. A transport action can only be started if the robot is idle, the destination machine is empty, and there is a part waiting to be moved to the machine. The firing of the transitions *rsP1* and *rsP2* removes one token from *M1empty* or *M2empty*, respectively. This ensures that no additional part is moved to a machine where a part is already being processed.

2.2 FMS Modeling With Colored Petri Nets

Petri nets are a powerful tool for the modeling of flexible manufacturing systems, because properties like concurrency and conflict can be expressed in a very elegant way. Additionally, there is a straightforward correspondance between events and activities in the FMS and places and transitions of the Petri net model.

However, if the considered FMS is large and fairly complex, modeling with uncolored Petri nets becomes cumbersome, and the models tend to be unintelligible. The same is true for other application areas. To overcome this general problem, colored Petri nets (CPN) [14] have been developed. They combine a graphical representation and a well-defined semantics on a substantially higher level than uncolored Petri nets.

In the following, we will show how our example is modeled with a CPN. We will use a notation similar to the one defined in [14]. Nevertheless, we need to incorporate time in our model, and thus extended Jensen's definition of CPNs to the (timed) modeling power of GSPNs. For this reason, immediate and timed transitions are introduced.

The combination of stochastic time and colored (or high-level) nets has already been proposed before [16, 24] and lead to the definition of stochastic well-formed colored Petri nets (SWN) [10]. In SWNs, both timed and immediate transitions are allowed.

Before the CPN model of our example is presented, colored Petri nets with timed and immediate transitions are briefly defined.

A (non-hierarchical) generalized stochastic colored Petri net (GS-CPN) is an eleven-tuple:

$$(\Sigma, P, T, \Pi, A, N, C, G, \Lambda, E, I) \qquad (2)$$

where

Σ is a set of non-empty types, called color sets,

P is the set of places,

T is the set of transitions (with $P \cap T = \emptyset$),

Π is the priority function, which maps transitions into nonnegative natural numbers, representing their priority level (for timed transitions, the priority is equal to zero),

A is a set of arcs,

N is the node function which is defined from A into $P \times T \cup T \times P$ (similar to input and output arcs),

C is a color function, yielding for each place the corresponding color set,

G is the guard function which maps each transition into a marking-dependent expression evaluating to true or false. Transitions with a guard evaluating to false are disabled.

Λ maps transitions into positive real numbers. For timed transitions t_k, $\Lambda(t_k)$ is called the rate of the transition, i.e. the firing time of transition

t_k is exponentially distributed with parameter $\Lambda(t_k)$. If t_k is an immediate transition, i.e. it fires without delay, $\Lambda(t_k)$ denotes the firing weight, which is used to solve conflicts between immediate transitions.

E is an arc expression function. It maps each arc into the expression associated with the arc.

I denotes an initialization function, which for each place evaluates into the initial marking.

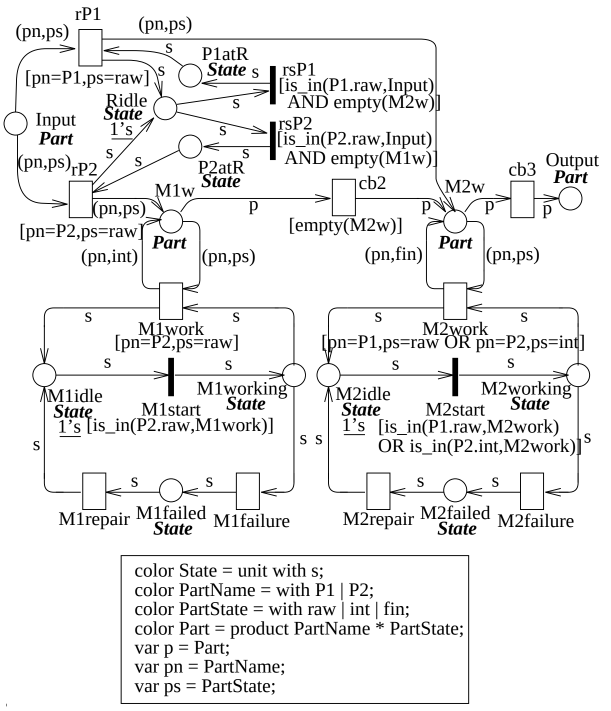

Fig. 4: Colored Petri Net model of the manufacturing cell

The GS-CPN model of the flexible manufacturing cell is shown in Fig. 4. In contrast to the GSPN model shown in the previous section, the transport line with the conveyor belts can be modeled with one place for each buffer. This is due to the possible differentiation between individual tokens.

An important part of a colored Petri net consists of the declaration of color sets and variables. Only previously defined variables and color sets may be used in the model. In [14], a language called CPN-ML (which is based on the language Standard ML) is used for these declarations and for the other net inscriptions as well.

In our example, the following declarations are made (see inside the box in Fig. 4):

State is a color set with only one color (similar to uncolored tokens),

PartName is a color set which describes the name of a product; allowed colors are *P1* and *P2*,

PartState is a color set describing the state of a part; allowed colors are *raw*, *int* (for intermediate), and *fin* (for finished),

Part is a color set describing a part; this set is a combination of the name and the state of a part,

p is a variable of type *Part*,

pn is a variable of type *PartName*, and

ps is a variable of type *PartState*.

Using these declarations, the manufacturing cell can be modeled. Each place has as inscriptions the name and the color set of the place (corresponding to the color of tokens which may reside in this place; the name of the color set is set in bold italics). Additionally, for some places the initital marking is defined. For instance 1's written near place *M1idle* means that this place is initialized with one token of type s. For places containing no token in the initial marking, the description of the initialization function is omitted. From the declarations, we know that s is an uncolored color set (*unit*), which was given the name s.

The inscriptions of transitions consist besides their names of an optional *guard*. A guard is a function which yields a boolean value; a transition is enabled in a given marking only if its guard evaluates to true. The boolean function *empty(*p*)* returns true if there is no token in place p. Thus transition *cb1* is enabled (the conveyor belt 1 may transport a part) only if place *M1w* is empty. Accordingly, the function *is_in(*t,p*)* evaluates to *true* if a token of color t is in a place p.

An expression is written beside each arc describing variables (or color constants) of tokens which can be transferred through them. When a transition is tested to be enabled, the variables of all arcs connected to this transition have to be bound to actual colors of the corresponding type. While doing this, variables with identical names have to be bound to the same colors. This might be possible in many different ways, and each of them is called a *binding*. When a transition is enabled under a certain binding, it may *occur* (fire).

For the two machines, the guards are more difficult. Transition *M1start* fires only if there is a token of type *P2.raw* in place *M1work*. Machine 1 can only fire if there is a token in place *M1working* and in place *M1w*. The latter has to be a raw part of type 2, because only in this case the guard evaluates to true. When machine 1 fires, this part is replaced by a part of type 2 with state *int*, and in place *M1idle* is a token representing the idle status of the machine.

The subnet modeling the second machine is very similar to the one described (only some guards are different), and thus we do not explain it further.

The robot is modeled with the transitions *rP1*, *rP2*, *rsP1*, and *rsP2*, and places *Ridle*, *P1atR*, and

P2atR. Their meaning was already described in section 2.1. The guard attached to transition *rsP1* ensures that this transition fires only if there is a token of color *P1.raw* in place *Input* and place *M2w* is empty.

2.3 Uncolored versus Colored Petri Nets

In this section we compare the use of colored and uncolored Petri nets for the modeling of flexible manufacturing systems. With the explanation of some of the advantages and disadvantages of both approaches, it is shown why a special method is better suited for this application area.

Petri nets in general combine a graphical description and a well-defined semantics in one formalism. Properties of FMS like concurrency and conflicts as well as blocking and shared resources can be modeled in a very natural way. Therefore, generalized and stochastic Petri nets [2] have been used many times for the modeling and performance evaluation of FMS [4, 5]. GSPNs are easy to use, their behaviour can be described almost purely graphically, and their timed analysis is well-established.

On the other hand, for FMS of more realistic size, the graphical representation gets very complex and hard to understand even for an experienced modeler. This problem was also encountered in section 2.1, although the example was very simple and small. Especially the modeling of buffers containing different parts leads to models in which the net structure does not correspond to the layout of the FMS anymore.

The natural choice to overcome this problem is the use of colored Petri nets, as it has been shown in section 2.2. Their higher level of modeling power, in particular the existence of individual tokens, results in much simpler models. Using CPNs, similar machines look the same, even if they process parts of different types.

The disadvantage of CPNs lies in the need to declare color sets and variables besides their graphical representation. To understand a CPN model, one has to read and understand the declarational part first, which can be quite complex. In addition, there are much more net inscriptions (such as arc expressions, guards and place color sets) needed in comparison to uncolored Petri nets. For a special application area like FMS with predefined color types many of the inscriptions could be omitted. Furthermore, if places and arcs of different color sets would differ in their graphical appearance, there would be no need to write down their corresponding color set.

When modeling a system with a CPN, the modeler is free to choose the balance between the graphical and the textual description of the net. In the extreme, any CPN model can be translated into a model with only one transition and one place, but with arbitrary complex net inscriptions [14]. If we want to have understandable models, this freedom has to be restricted.

In FMS, the capacity of part buffers is always limited. In our example, at each machine there can be only one part at a time. It would be comfortable for the modeler, if this property could be modeled directly, avoiding the extension of the guards of transitions which possibly add tokens to the limited places.

3 A Special Modeling Method for Flexible Manufacturing Systems

In this section we first define a modeling method which is especially tailored to the modeling of FMS. Afterwards, the corresponding graphical representation is introduced. As an example, the FMS cell is modeled using the proposed method subsequently.

3.1 Definition of the Special Method

Based on the considerations of the previous section, we define a modeling method for flexible manufacturing systems. We try to incorporate the advantages of colored and uncolored Petri nets for this special application area of Petri nets.

A special Petri net for the modeling of FMS is a nine-tuple:

$$(P, T, \Pi, A, G, \Lambda, E, I, C) \tag{3}$$

where

P is the set of places, $P = P^P \cup P^S$, $P^P \cap P^S = \emptyset$, where P^P is the set of places of type *Part*, and P^P is the set of places of type *State*. Tokens in places of type *State* are uncolored, while the tokens in places of type *Part* have the form <PartName>.<PartState> . *PartName* denotes the name, while *PartState* corresponds to the state of the part which is modeled by a token of the color set *Part*.

T is the set of transitions $T = T^T \cup T^I$, $T^T \cap T^I = \emptyset$, and $P \cap T = \emptyset$. T^T denotes the set of timed transitions, and T^I the set of immediate transitions.

Π is the priority function, which maps transitions into nonnegative natural numbers, representing their priority level (for timed transitions, the priority is equal to zero).

A is a set of arcs, $A \subseteq P \times T \cup T \times P$ (the input and output arcs).

G is the guard function which maps each transition into a marking-dependent expression evaluating to true or false. Transitions with a guard evaluating to false are disabled.

Λ maps transitions into positive real numbers. For timed transitions t_k, $\Lambda(t_k)$ is called the rate of the transition, i.e. the firing time of transition t_k is exponentially distributed with parameter $\Lambda(t_k)$. If t_k is an immediate transition (i.e. it fires without delay), $\Lambda(t_k)$ denotes the firing weight.

E is an arc expression function. It maps each arc into the expression associated with the arc.

I denotes an initialization function, which evaluates into the initial marking.

C is the capacity function, which yields for each place the maximum number of tokens. For unlimited places, C evaluates to ∞.

The main difference between CPNs and the given definition is the restriction of the color sets allowed for places. This is done by the pre-definition of color sets for places modeling states of the system and those modeling buffers (or other storage facilities) containing parts. As it will be shown later, the declarations of CPNs are then no longer needed. Therefore, the new method allows the easy modeling of complex systems with as much information coded in the graphical representation as possible.

The balance between the net inscriptions and the graphical model parts is now fixed, because the modeler is forced to use places of type *Part* for the modeling of storage facilities. Additionally, a maximum capacity can be associated with places, thus naturally modeling buffers with limited capacity. The further analysis (i.e. the generation of the reachability graph) of the net is changed by this extension only at the point where the enabling of transitions is tested. If the capacity of a place would be exceeded by the firing of a transition, the transition is disabled.

FMS models within this formalism are an image of the functional FMS layout. Each buffer is modeled with one place. Additionally, states and state changes of subsystems (manufacturing cells, robots) can be expressed with places containing uncolored tokens.

The determination of different place types was also proposed by Beck and Krogh [6], but in the context of uncolored Petri nets. They distinguish between *resource state places* and *operation places*, corresponding to the states of resources and operations in progress, respectively. Based on these considerations, they defined a modeling method called *modified Petri nets*. This Petri-net variation is then used for the synthesis of live and safe Petri net models of FMS.

3.2 Graphical Representation

As mentioned in the previous section, we use predefined color sets corresponding to two different types of places. To distinguish between them, different graphical representations are chosen. The circles denoting a place of type *State* are drawn thin, while for places of type *Part* they are drawn thick. Therefore, the color set attached to a place can be recognized from the graphical description. When using a color monitor, different drawing colors may also be used to differentiate between the two types of places and arcs.

Each arc is connected to exactly one place, and obviously the color set of the arc expression must comply with the color set defined for the place. Thus, the color set of each arc can be directly adopted from the connected place. For a better understanding of models, the arcs are also drawn thick or thin, according to their associated color set.

With these two simple changes, the net becomes much more readable, not only because the color set definitions of places can be omitted, but also due to the better visibility of the underlying structure. The use of (user-defined) different graphical representations for places and arcs has also been proposed in [14], but was only used for improving the readability and not as a general principle.

Shapes of transitions are chosen equivalently to GSPNs: timed transitions are denoted by a rectangle, and immediate transitions by a thick line. Near the transition, its name, priority and delay is written. For immediate transitions, the delay is replaced by a firing weight. An optional guard can be added within square brackets.

In Petri nets with colored tokens, transitions can be enabled only under a certain *binding*. To analyze such a binding, all variables contained in arc expressions connected to the transition have to be bound to actual colors. In CPNs, variables have to be defined to form arc expressions and guards.

When the modeling method introduced in this paper is used, the color set of each arc expression is known (see above). Therefore, variables don't have to be declared, because their color set is implicitly given by the color set of the arc expression they occur in. For instance, if an expression $a.b$ is attached to an arc which is connected to a place with color set *Part*, obviously a corresponds to the name and b to the state of the part.

For many cases, the arc expressions can be simplified even more. If the color of a token is not altered by the firing of a transition (i.e. it is only moved to another place), the arc expression can be omitted. Or, if only parts named *P2* may be moved through an arc, it is quite forward to write $P2.*$ as the arc expression (using $*$ as a wildcard symbol).

The allowed part names and states can be defined with a description of the product routes similar to the one which was given for our example in section 2.

From these considerations it is clear that the declaration of colors and variables can be left out, thus avoiding the need to understand source-code like text before looking at the actual net model.

Tokens of both color sets are represented by black dots as it is known from uncolored Petri nets. To distinguish between different parts, their names and current states can be added in the graphical representation.

If a place has a limited capacity, the corresponding number is written near the place and enclosed with square brackets.

3.3 FMS Modeling with the Special Method

This section comprises the modeling of our example, the FMS cell, with the new method defined and explained before.

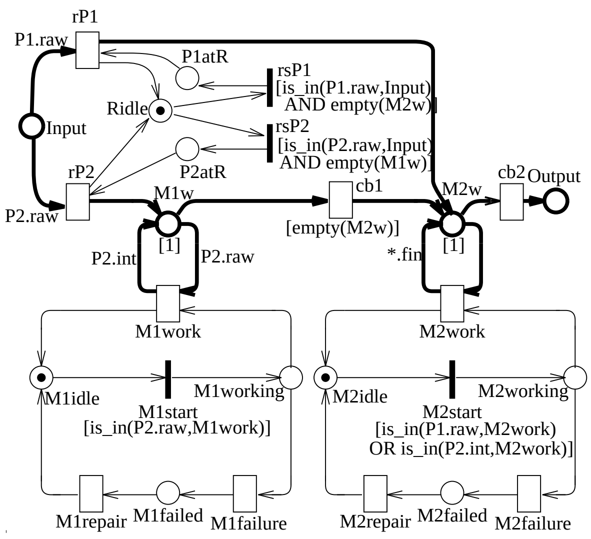

Fig. 5: Model of the manufacturing cell with the new method

The model of the manufacturing cell using the new method is shown in Fig. 5. The graphical structure is the same as the one of the GS-CPN model, but there are considerably fewer net inscriptions. Additionally, the box containing the declarations is omitted.

All names are equal to those used in the CPN model in section 2.2. Because the behavior of the model is the same as well, we refer to the explanation given in the mentioned section. In the following, only differences which are due to the use of the new modeling method are described.

The structure of the model is clearly visible, as one can distinguish between the transport line and the modeling of the states of the machines and the robot.

The guards attached to the transitions *cb*<i> of the CPN model could be omitted because now the limited capacity of the places where the processing is done is defined at the places.

There is no need to write down the color of the tokens of type *State* now, and therefore the initialization functions of place *M1idle* and *M2idle* are no longer necessary. The tokens are simply drawn inside the places instead, as one is used to from uncolored Petri nets. For the same reason there are no arc expressions needed for arcs of color set *State*.

The guards of the transitions *M1work* and *M2work* are now implemented in the arc expressions. The arc expressions for the first machine mean that machine 1 can only process parts of type 2, which are in state *raw*. The state of those parts is changed by the machine to *int*.

The second machine processes parts of any type and state which may appear in place *M2w*, which in fact are only parts of the colors *P1.raw* and *P2.int* . There are two possible bindings of transition *M2work* according to these two colors. After one of those parts is processed by machine 2, the part state is changed to *fin*. The wildcard symbol $*$ denotes that there is no change in the part type. For arcs with omitted expression one could also write $*.*$ (for any part type in any state).

4 Conclusions

In this paper we discussed different kinds of Petri nets and their application to the modeling of flexible manufacturing systems. We found that both the use of colored and uncolored approaches leads to different advantages and disadvantages.

From these considerations, a new modeling method based on colored Petri nets is introduced, which is especially tailored for the modeling of FMS. Models of manufacturing systems using the proposed method are better understandable because state changes and the flow of products through the system are modeled differently. Although the high modeling power of colored Petri nets is exploited, no definitions of color sets, variables etc. is needed. The impact of the use of the FMS modeling method is shown by an example.

The proposed method of defining a special modeling environment for a given application area is not limited to the field of FMS. Instead, the development of specialized CPN-based modeling methods may lead to a broader acceptance of Petri net models in areas like telecommunication and computer systems.

References

[1] Ajmone Marsan, M.: Stochastic Petri Nets: An Elementary Introduction. In: G. Rozenberg (Ed.), Advances in Petri Nets 1989, Lecture Notes in Computer Science **424**, Springer 1990, 1–29

[2] Ajmone Marsan, M., Balbo, G., Conte, G.: A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems. ACM Trans. on Computer Systems **2** (1984) 93–122

[3] Al-Jaar, R.Y., Desrochers, A.A.: Petri Nets in Automation and Manufacturing. In: G.N. Saridis (Ed.), Advances in Automation and Robotics **2** Knowledge-based Systems for Intelligent Automation, JAI Press (1990)

[4] Al-Jaar, R.Y., Desrochers, A.A.: Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri Nets. IEEE Trans. on Robotics and Automation **6** 1990

[5] Balbo, G., Chiola, G., Franceschinis, G., Molinar Roet, G.: Generalized Stochastic Petri Nets for the Performance Evaluation of Flexible Manufacturing Systems. Proc. Int. Conf. on Robotics and Automation, Raleigh, North Carolina (1987) 1013–1018

[6] Beck, C. L., Krogh, B. H.: Models for Simulation and Discrete Control of Manufacturing Systems. Proc. Int. Conf. on Robotics and Automation (1986) 305–310

[7] Bruno, G., Morisio, M.: Petri-Net Based Simulation of Manufacturing Cells. Proc. Int. Conf. on Robotics and Automation (1987) 1174–1179

[8] Carrasco, J. A.: Automated Construction of compound Markov chains from generalized stochastic high-level Petri nets. Proc. 3rd Int. Workshop on Petri Nets and Performance Models, Kyoto, Japan (1989) 93–102

[9] Chiola, G., Ajmone Marsan, M., Balbo, G., Conte, G.: Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications. IEEE Trans. on Software Engineering **19** (1993) 89–107

[10] Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. IEEE Trans. on Computers **42** (1993) 1343–1360

[11] Ciardo, G., German, R., Lindemann, C.: A Characterization of the Stochastic Process Underlying a Stochastic Petri Net. Proc. 5th Int. Workshop on Petri Nets and Performance Models, IEEE Trans. on Software Engineering **20** (1994) 506–515

[12] Gentina, J.C., Corbeel, D.: Coloured Adaptive Structured Petri-Net : A Tool for the Automatic Synthesis of Hierarchical Control of Flexible Manufacturing Systems. Proc. IEEE Int. Conf. on Robotics and Automation, (1987) 1166–1173

[13] German, R.: Analysis of Stochastic Petri Nets with Non-Exponentially Distributed Firing Times. Ph. D. Dissertation, Department of Computer Science, Technische Universität Berlin (1994)

[14] Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. **1** EATCS Monographs on Theoretical Computer Science, Springer 1992

[15] Kasturia, E., DiCesare, F., Desrochers, A. A.: Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets. Proc. Int. Conf. on Robotics and Automation (1988) 1114–1119

[16] Lin, C., Marinescu, D. C.: On Stochastic High-Level Petri Nets. Proc. Int. Workshop on Petri Nets and Performance Models (1987) 34–43

[17] Lindemann, C., Ciardo, G., German, R., Hommel, G.: Performability Modeling of an Automated Manufacturing System with Deterministic and Stochastic Petri Nets. Proc. IEEE Int. Conf. on Robotics and Automation, Atlanta, Georgia, USA (1993) 576–581

[18] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proc. of the IEEE **77** (1989) 541–575

[19] Narahari, Y., Viswanadham, N.: A Petri Net Approach to Modeling and Analysis of Flexible Manufacturing Systems. Annals of Operations Research **3** (1985) 449–472

[20] Silva, M., Valette, R.: Petri Nets and Flexible Manufacturing. In: G. Rozenberg (Ed.), Advances in Petri Nets 1990, Lecture Notes in Computer Science **424**, Springer (1991) 374–417

[21] Viswanadham, N., Narahari, Y.: Coloured Petri Net Models for Automated Manufacturing Systems. Proc. Int. Conf. on Robotics and Automation, Raleigh, North Carolina (1987) 1985–1990

[22] Viswanadham, N., Narahari, Y., Johnson, J.: Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Sytems. IEEE Trans. Robotics and Automation **6** (1990) 713–723

[23] Watson, J.F., Desrochers, A.A.: Applying Generalized Stochastic Petri Nets to Manufacturing Systems containing Non-exponential Transition Functions. IEEE Trans. System, Man, and Cybernetics **21** (1990) 1008–1017

[24] Zenie, A.: Colored Stochastic Petri Nets. Proc. Int. Workshop on Petri Nets and Performance Models (1985) 262–271

[25] Zhou, M., McDermott, K., Patel, P. A.: Petri Net Synthesis and Analysis of a Flexible Manufacturing System Cell. IEEE Trans. on Systems, Man, and Cybernetics **23** (1993)

[26] Zurawski, R. Z., Dillon, T. S.: Systematic Construction of Functional Abstractions of Petri Net Models of Typical Components of Flexible Manufacturing Systems. Proc. Int. Workshop on Petri Nets and Performance Models (1993) 248–257